UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/776,909	02/11/2004	Louis R. Degenaro	YOR919990064US2 (8728-258	3057

46069 7590 10/20/2009

F. CHAU & ASSOCIATES, LLC
130 WOODBURY ROAD
WOODBURY, NY 11797

EXAMINER
VERBRUGGE, KEVIN

ART UNIT	PAPER NUMBER
2189	

MAIL DATE	DELIVERY MODE
10/20/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/776,909	DEGENARO ET AL.	
	Examiner	Art Unit	
	Kevin Verbrugge	2189	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-51* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-51* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This non-final Office action is in response to the appeal brief filed 5/18/07 appealing from the final Office action mailed 10/18/06. This non-final Office action corrects the inadvertent omission of any treatment of claims 49-51 and the inadvertent omission of a rejection of claims 35-40 under 35 U.S.C. 101 in the Examiner's Answer mailed 11/1/07. The Examiner's Answers mailed 9/21/07 and 11/1/07 are hereby vacated.

This case has been assigned to the Examiner below since the previous Examiner has joined the Central Reexam Unit.

In view of the "Order Returning Undocketed Appeal to the Examiner" mailed on 6/25/09, PROSECUTION IS HEREBY REOPENED. New grounds of rejection are set forth below.

To avoid abandonment of the application, Applicant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) initiate a new appeal by filing a notice of appeal under 37 CFR 41.31 followed by an appeal brief under 37 CFR 41.37. The previously paid notice of appeal fee and appeal brief fee can be applied to the new appeal. If, however, the appeal fees set forth in 37 CFR 41.20 have been increased since they were previously paid, then Applicant must pay the difference between the increased fees and the amount previously paid.

A Supervisory Patent Examiner (SPE) has approved of reopening prosecution by signing at the end of this Office Action.

Claims 1-51 are pending and are herein rejected.

Remarks

It is noted that no prior art rejection has been applied to claims 12-13 and 28-30. These claims are only rejected in view of the double patenting rejection.

The summary of claimed subject matter contained in the brief is generally correct. However, because of Applicants' apparent misunderstanding (as evidenced by arguments regarding claims 42 and 46) of the relationship between a probability of a statement execution and a likelihood of a value of a cacheable entity changing, which is directly related to the desirability of performing at least one cache transaction, the Examiner will attempt to provide additional explanations regarding the claim limitations "probability that at least one statement will execute", "desirability of performing at least one cache transaction", and "likelihood of a value of a cacheable entity changing" to clarify the relationship among them for easier understanding of the claims and teachings of references as related to the claimed limitations.

First of all, the Examiner notes that Applicants use the terms "probability" to represent "likelihood" (see specification, page 36, lines 10-11, "a probability is determined which represents the likelihood that the detected statement will be executed"). The term "probability" seems to be used as a synonym for "likelihood".

"A **probability that at least one statement will execute**" depends on the structure of the program being analyzed. As Applicants explained (Appeal Brief, page 3, last sentence), a statement that is outside a conditional branch will execute unconditionally with probability of 1 (certainty). On the other hand, a program statement within a conditional branch may or may not

execute depending on which of the two possible branches is taken by the execution of the program. The probability that a statement in one branch will execute is p, where p is the probability that the program execution will take this branch. The probability that a statement in the other branch will execute is 1-p. The Examiner notes that Applicants carefully avoided the use of the word "calculating" and instead consistently used a broader term "determining" in the specification and the claims. Therefore, Applicants do not disclose, nor do they claim, calculation of a probability value. The disclosed and claimed determination seems to be an assessment of whether it is likely to take one branch or the other branch.

"A **desirability of performing at least one cache transaction**" depends on whether the value of a cacheable data object will change and whether the data object is likely to be accessed in the near future. A cache is a high-speed memory (faster than the main memory) with relatively small capacity (because it is more expensive than slower main memory) used in a computer system to temporarily keep data items likely to be accessed again to improve memory access speed. In a typical caching system, when a data item is to be accessed the system searches the cache memory. If the data item needed is in the cache memory, it is retrieved from the faster cache (this is called a cache "hit" in the art of caching). If not, the system must retrieve the data from the slower main memory (a cache "miss"). Keeping only data items likely to be accessed again improves the overall system performance. Therefore, it is not desirable to keep data items used only once or infrequently accessed in cache because they occupy limited space that should be populated with more frequently accessed data. It is also not desirable to keep obsolete data or a data item that is about to be replaced with a new value. Caching systems use various known techniques, such as least recently used, least frequently used, and other cache

replacement algorithms to attempt to replace infrequently accessed data with fresh data. These systems also use cache invalidation techniques to flag obsolete data for eviction or "flushing" to make room for more desirable data. Some systems try to anticipate the likelihood of needing a data item, by analyzing access patterns or program structures, and populate the cache (i.e., make caching decisions) prior to the actual need for the data.

As Applicants explained (Appeal Brief, page 4), if a data item is likely to change its value 1) it is not desirable to populate this data item in cache if it is not in cache already, and 2) it may be desirable to invalidate if it is occupying space in cache because it is likely to become obsolete. Thus, one factor that affects the desirability of a caching transaction is whether the **value of a data object is likely to change,** which depends on the actual statement (other factors include whether a data object is likely to be needed soon or whether the object is likely to be infrequently accessed). For example, in a statement X = Y + Z, the value of X is likely to change, if executed, depending on the values of Y and Z. On the other hand, values of Y and Z will not change by the execution of this statement. Whether the value of X may change depends in turn on whether the statement will execute at all. If the statement is not executed, the value of X will not change. Therefore, the likelihood of the value of X changing depends on both the statement (structure of the statement) itself and whether the statement will execute (structure of the program, i.e., probability that the program will take the branch that contains the statement).

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-3, 5, 8-11, 14, 35-40, 42, 49, and 51 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

With respect to claims 1-3, 5, 8-11, 14, 42, and 49, the language of the claims raises a question as to whether the claims are directed merely to an abstract idea that is not tied to a technological art, environment or machine which would result in a practical application producing a concrete, useful, and tangible result to form the basis of statutory subject matter under 35 U.S.C. 101. Even when automated, the claims seem to be directed to a method of planning/preparing with no practical application of the plan. This still amounts to a machine-manipulated abstract idea, which is considered non-statutory.

Further regarding claims 35-40 and 51, the claims are directed to a system, but the specification explicitly teaches that the system of the invention could be implemented entirely in software (see page 6, lines 2-7 and 21-24). Since software per se does not fall within one of the four statutory categories of invention, the claims are directed to non-statutory subject matter and are therefore not patentable.

Double Patenting

The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

Claim 1-37 of U.S. Patent 6,725,333 contains every element of claim 1-51 of the instant application and as such anticipates claim 1-51 of the instant application.

"A later patent claim is not patentably distinct from an earlier patent claim if the later claim is obvious over, or **anticipated by**, the earlier claim. In re Longi, 759 F.2d at 896, 225 USPQ at 651 (affirming a holding of obviousness-type double patenting because the claims at issue were obvious over claims in four prior art patents); In re Berg, 140 F.3d at 1437, 46 USPQ2d at 1233 (Fed. Cir. 1998) (affirming a holding of obviousness-type double patenting where a patent application claim to a genus is anticipated by a patent claim to a species within that genus). " ELI LILLY AND COMPANY v BARR LABORATORIES, INC., United States Court of Appeals for the Federal Circuit, ON PETITION FOR REHEARING EN BANC (DECIDED: May 30, 2001).

As to the newly added limitations, they are implicit in the patented claims.

Claim Rejections - 35 USC § 112

The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

Claims 42-44 and 46-48 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the

relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

With respect to claims 42 and 46, according to the specification the probability of a value of a cacheable entity changing is based on the probability that a statement will execute, not the other way around as claimed (see specification page 36, lines 14-23, the probability of a value of a cacheable entity changing depends on the probability that the statement containing the cacheable entity will execute because if the statement is not executed the value cannot change; a change in value can only be caused by the execution of the statement, and whether the statement will execute depends on the structure of the program, i.e., whether the program will take the branch containing the statement, and not on the whether the statement will change some value when executed).

At page 5, lines 10-14, the specification states "a probability is determined which represents the likelihood that the detected statements will be executed (i.e., the likelihood that one or more cacheable entities will change due to execution of the statement) (step 601)." When read in isolation, this statement seems to suggest that the probability of execution is the same as the likelihood that a cacheable entity will change, but when read in conjunction with the preceding sentence (page 5, lines 5-10) and figure 6, it is fairly clear that the specification is consistent with the Examiner's understanding that the likelihood that one or more cacheable entities will change depends on both the probability that the statement containing the cacheable entity will execute (program structure, spec. page 5, lines 10-20) and that the statement is structured to modify the value of the cacheable entity (statement structure, spec., lines 5-10). See section (5) Summary of Claimed Subject Matter above.

With respect to claims 43, 44, 47 and 48, caching, updating and invalidating are to be performed depending on the likelihood of a value of a cacheable entity changing exceeding or not exceeding a threshold, as opposed to the probability of a statement execution exceeding or not exceeding a threshold (see specification page 37, lines 1-15).

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1, 5, 10, 18, 22, 27, 35, 39, 40, 41, 45 and 49-51 are rejected under 35 U.S.C. 102(b) as being anticipated by Nakanishi *et al.* (US Patent No. 5,940,857, hereinafter "Nakanishi").

Nakanishi discloses an automated method for managing a plurality of cacheable entities, comprising the steps of:

analyzing program code to determine if there is at least one statement which can affect a desirability of performing at least one cache transaction, if the at least one statement is executed (abstract, col. 4, lines 22 – 44, an instruction is analyzed to determine whether it is desirable to cache the instructions in the next block);

augmenting the program code with additional code to assist in determining the desirability of performing the at least one cache transaction (col. 27, lines 42-47, "advance read instruction"),

determining a probability that the at least one statement will execute (col. 27, lines 48-55, branch destination block with high possibility of being read out from the main memory, i.e., high probability of execution, is determined and indicated with an advance read instruction);

determining the desirability of performing the at least one cache transaction based on a probability that the at least one statement will execute (see above, see also col. 6, lines 43 – 50) .

As per claims 49-51, Clearly Nakanishi's device will perform the claimed cache transaction if it is desirable to do so.

Claims 1, 2, 4 – 6, 8, 18, 19, 21, 23, 25, 35, 36, 38 – 41, 45 and 49-51 are rejected under 35 U.S.C. 102(b) as being anticipated by Dubey (US Patent No. 5,774,685).

With respect to claims 1, 41, 4, 5, 6, 8, 18, 45, 21, 23, 25, 35, 38, 39, 40, 41, and 45, Dubey discloses a system for managing a plurality of cachable entities, comprising:

a program analyzer to analyze program code and determine if there is at least one statement which may affect a desirability of performing at least one cache transaction, if the at least one statement is executed (col. 3, lines 58 – col. 4, line 4);

the program analyzer determining a probability that the at least one statement will execute (abstract, speculative prefetches associated with conditional braches by its very nature is a determination that it is probable that a particular branch will be taken) and determining the desirability of performing the at least one cache transaction based on a probability that the at least one statement will execute (col. 4, lines 31 – 44, the compiler determines the desirability of prefetching data based on a speculation that the prefetched data will actually be needed, i.e.

based on the probability or likelihood that the branch that requires the data will be taken, and inserts a STOUCH instruction a prefetch point) determines; and

a cache manager for performing the at least one cache transaction if it is determined to be desirable (col. 2, lines 21 – 27, instructions identified by the STOUCH instruction, i.e., determined to be desirable, are fetched in the instruction cache).

With respect to claims 2, 19 and 36, the desirability of performing the at least one cache transaction is based on one of a frequency of access of at least one cachable entity (col. 13, lines 19 – 25, the cache transaction involves caching the replacing the old cached data with the prefetched data, the replacement strategy is based on the least recently used algorithm, i.e. least frequently used), a size of at least one cachable entity, a time to one of fetch and materialize at least one cachable entity, a lifetime of at least one cachable entity (LRU data is also the oldest data in the cache), and a combination thereof.

As per claims 49-51, clearly Dubey's device will perform the claimed cache transaction if it is desirable to do so.

Claims 1 – 8, 10, 14 – 25, 27, 31 – 51 are rejected under 35 U.S.C. 102(b) as being anticipated by Cytron *et al.* (Automatic Management of Programmable Caches, Proceedings of the 1988 International Conference on Parallel Processing, 1988, pp. 75 – 84).

With respect to claims 1, 41, 4, 5, 6, 8 18, 45, 21, 23, 25, 35, 38, 39, 40, 41 and 45, Cytron discloses a system for managing a plurality of cachable entities, comprising:

a program analyzer (page 228, right column, second paragraph) to analyze program code and determine if there is at least one statement which may affect a desirability of performing at

least one cache transaction, if the at least one statement is executed (page 231, 2.0 Algorithms, the algorithm analyzes a program to determine cacheability of variables, see page 230, left hand column, 1.1 Software-Controlled Caches for definition of cacheability or degree of desirability of caching, see also page 232, 2.2 Posting Values to Global Memory, Cyclon discloses a program code sequence that should cause, i.e. highly desirable, a posting of a cached value, i.e. cache transaction);

the program analyzer determining a probability that the at least one statement will execute (1.2 Execution model, the program analyzer of Cytron's disclosure analyzes DO loops to achieve parallelism, statements in a DO loop are always executed, i.e., probability of execution is determined to be 1, see also 2.1, Processor-Crossing dependencies, the analyzer also determines a probability that a statement will execute in different processors) and the desirability of performing the at least one cache transaction based on a probability that the at least one statement will execute (figure 4, the probability of executing a Write(P_i,X) -> Read(P_j,X) pattern in a DO loop shown in figure 4 is 1, and the analyzer determines the desirability by augmenting the code with a POST operation code); and

a cache manager for performing the at least one cache transaction if it is determined to be desirable (this is accomplished when the augmented program in figure 4 is executed).

With respect to claims 2, 19 and 36, the desirability of performing the at least one cache transaction is based on one of a frequency of access of at least one cachable entity, a size of at least one cachable entity, a time to one of fetch and materialize at least one cachable entity, a lifetime of at least one cachable entity (cacheability is determined based on an expected lifetime,

for example, expected lifetime for a non-cacheable entity is zero and expected lifetime for a cacheable entity is greater than a temporary cacheable entity), and a combination thereof.

With respect to claim 3, 20, and 37, wherein the at least one statement is a statement that modifies a value of at least one cachable entity (a write statement modifies a value the variable being written), and wherein the desirability is based on an expected lifetime of the at least one cachable entity (see the rejection of claim 2 above).

With respect to claims 7 and 21, at least one of the step of invalidating the at least one cachable entity stored in the cache and the step of updating the at least one cachable entity stored in the cache comprise the step of performing data update propagation (DUP) (figure 4, updated X(j) is propagated to the next iteration, see also figure 3, a cached value from one processor is updated and propagated to another processor).

With respect to claims 10 and 27, the cachable entities include query results (a processor querying the cache for a data item reads on this claim).

With respect to claims 14, 15, 31 and 32, the at least one statement is a type that one of creates at least one cachable entity, deletes at least one cachable entity, and modifies a value of at least one cachable entity (figure 4), wherein the analyzing step comprises the steps of:

generating an invalidation key format in accordance with the type of the at least one statement (figure 5); and

augmenting the program code with additional code for calculating an invalidation key in accordance with the generated invalidation key format (figure 5).

With respect to claims 16 and 33, wherein the step of invalidating at least one cachable entity comprises one of purging the cachable entity from the cache, purging the cachable entity

from the cache and repopulating the cache, and updating the cache (page 232, 2.3 Invalidating Cache, see also page 233, 2.4 Flush).

With respect to claims 17 and 34, the step of performing at least one cache transaction comprises the step of initializing a cache (Cytron disclose a software controlled cache which must be initialized and populated to function properly, initial entries of values meet the limitations of this claim).

With respect to claims 42 and 46, Cytron discloses determining said probability based on likelihood of a value of a cachable entity changing (a write statement changes the value of X, see figure 4).

With respect to claims 43, 44, 47, and 48, the values are cached, posted and invalidated regardless of the threshold value, therefore, they are cached and invalidated if the probability is less than a threshold and they are cached and invalidated if the probability is greater than a threshold. See page 232.

With respect to claims 49-51, clearly Cytron's device will perform the claimed cache transaction if it is desirable to do so.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 9 and 26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cytron in view of Levine *et al.* (US Patent No. 6,073,129).

Cytron discloses all of the limitations of the parent claims as discussed above. However, Cytron does not specifically disclose the use of SQL code and that the code includes a SET statement. On the other hand Levine specifically discloses RDMS system that uses SQL codes including a SET statement (col. 29, line 57).

It would have been obvious to one of ordinary skill in the art, having the teachings of Cytron and Levine before him at the time the invention was made, to use the RDMS teachings of the computer system with caching of Levine in the computer system with caching of Cytron, in order to in apply Cytron's automatic management of programmable caching method in a practical and widely used real life application such as the one disclosed by Levine.

Response to Arguments

Rejections under 35 U.S.C. 101

Claim 1 is directed to a method for managing a plurality of cacheable entities. Yet the claimed steps of this "method of managing" cacheable entities only comprise analyzing a program code, determining a probability and a desirability. Not even a single step directed to the actual management of cacheable entities is claimed by Applicant. The claim does not even require any cacheable entity that the method is supposed to manage. In the absence of any step that actually uses the results of analysis and determinations for managing cacheable entities, for example by making caching decisions based on the results, analysis of code and determination of probabilities amount to automation of algorithmic steps that do not impart any useful practical

and concrete results by themselves. Algorithms that are not put to any practical use are not statutory even when they are automated. A method for managing cacheable entities that does not recite any step to actually manage cacheable entities and does not even require any cacheable entity that the method is supposed to manage cannot produce any useful, tangible, and practical result of managing cacheable entities.

Double Patenting Rejections

Applicant concedes the propriety of double patenting rejections.

Rejections under 35 U.S.C. 112, 1st paragraph

Claims 42 and 46

Claims 42 and 46 recite the limitation "determining said probability based on a likelihood of a value of a cacheable entity changing." "[S]aid probability" refers to the "probability that the at least one statement will execute." Applicant contends that this limitation is supported by passage in the specification at page 36, lines 17-23. However, the passage actually supports the rejection. According to the passage, if the probability that a statement x = a * b will execute is high, the compiler would conclude that x has a high probability of changing. In other words, the likelihood that the value of x (a cacheable entity) will change depends on the probability of the statement executing, not the other way around as claimed. The value of x cannot change if the statement is not executed. The value of x has no effect on whether the statement containing the variable x will execute as claimed, because the execution of the statement containing the variable

x depends on the structure of the program and other variables (in the cited passage above, the value of y) that affect branching decisions. On the other hand, the probability that the statement containing the variable x will execute will always affect the value of x, because the value of x will be recalculated every time the statement is executed. Applicant is confused about the causal relationship between the probability that a statement containing a variable will execute and the likelihood that the value of the variable in the statement will change.

Claims 43, 44, 47, and 48

These claims depend from claims 42 and 46. Therefore, they inherited the logical flaw of their parents discussed above. In addition, the claims require caching, invalidating, or updating a value of said cacheable entity, if said probability is greater than or equal to a threshold. Applicant seems to be conflating the probability of a statement executing and a likelihood of a cacheable entity changing, which are two separate values. The claims require making certain caching decisions based on the result of comparing the probability of a statement executing with a threshold. This is not supported by the specification. At page 37, lines 1-15, the specification clearly discloses that caching decisions are made based on the probability or likelihood of the value of the cacheable entity changing. The caching decisions are not made based on the result of comparing the probability that a statement will execute with a threshold as claimed.

Rejections under 35 U.S.C. 102

Nakanishi Reference

Applicant asserts that Nakanishi does not disclose "analyzing program code to determine if there is at least one statement which can affect a desirability of performing at least one cache transaction, if the least one statement is executed." At col. 4, lines 22-44, Nakanishi clearly discloses this claimed limitation. Nakanishi discloses that instructions in a program block is analyzed to predict if it will be necessary to read in the next block from the main memory, and if so to cache the next block to reduce the probability of cache miss (Nakanishi calls this cache "error"). Applicant contends that this finding is fundamentally flawed for various reasons without articulating those reasons by clearly discussing why Nakanishi's teachings do not read on the language of the claim. Applicant's only real argument seems to be that the purpose of Nakanishi's analysis of program code is not the same as Applicant's, which is irrelevant in anticipation analysis.

Applicant does not dispute that Nakanishi discloses program code analysis. There is no dispute that this analysis of the program code is performed to decide whether another block is to be speculatively read into a cache prior to its actual need. Applicant admits that this decision is based on determination of presence of a branch instruction and its predicted outcome (Appeal Brief, page 17, Nakanishi col. 18, lines 56-65). Applicant also admits that Nakanishi teaches that "[t]he instruction analysis section (5) analyzes the content of a read block of instruction to determine if there is a branch instruction or not, and if there is a branch instruction, and branch destination address is calculated (see e.g., Col. 19, lines 12-20)" (Appeal Brief, page 17, 1st full paragraph, last sentence). In one embodiment of the Nakanishi's invention, the decision to speculatively cache the next block ("at least one cache transaction") depends on the presence of a branch instruction ("if there is at least one statement") and whether resulting destination address

is outside the region ("desirability of performing at least one cache transaction", not desirable to cache the next block if it is outside the region) of the current and the next block (see col. 2, lines 40-46). Nakanishi clearly discloses all of the elements of the limitation.

Applicant further argues that "the Examiner fails to address the specific claim language that the program code is analyzed to determine if there is at least one statement which can affect the desirability of performing at least one cache transaction, if the at least one statement is executed" (Appeal Brief, page 16, last paragraph through page 17, first paragraph). Applicant argues that "if the detected statement is not executed, it does not affect a desirability of performing a transaction" and that "[t]he Examiner does not explain how Nakanishi teaches this aspect of the claimed invention." This is a frivolous argument. Applicant is arguing limitations that are not in the claims. The claims do not specify what happens when the detected statement is not executed. While claims are interpreted in light of the specification, limitations are not imported from the specification.

Applicant also seems to be arguing that Nakanishi does not teach the claimed step of "determining a probability that the at least one statement will execute" without articulating in a reasonable and a convincing manner as to why this is so. Nakanishi discloses this throughout the specification, but claim 8 (col. 30 lines 41-52) summarizes this aspect of the disclosure. The claim states in part "branch predict means for predicting whether the branch instruction is more likely than not to execute a branch operation." In other words, the branch prediction analysis circuit determines whether the probability of executing a particular branch operation is greater than 0.5 (50 %). Thus the analysis means determines the probability that the instructions at the destination of the branch operation will execute. The same analysis also determines the

complementary probability that the instructions following the branch statement (i.e., instructions to be executed if decision is made not to perform a branch operation) will or will not execute.

Dubey Reference

Prefetching is a techniques used to improve memory access performance (see col. 1, lines 14-21). In a prefetch, a data item or an instruction is fetched into the cache (which is a faster access memory than the main memory) for possible future use prior to its actual need. This practice improves memory access speed by supplying the data needed from the cache and avoids processor wait time associated with slow main memory access cycles (because the needed data item is already in the faster cache memory). Dubey teaches a method of speculatively prefetching objects into a cache and making other caching decisions, such as discarding or retaining prefetched cached data, based upon evaluation of speculative conditions (see title of the invention and abstract).

Applicant's arguments seem to depend entirely on the anticipating Dubey reference not using the same words used by Applicant in the claims rather than the substance of the disclosure.

For example, Applicant admits that Dubey teaches that compile time analysis is performed to determine locations within a program wherein cache misses are likely to occur at run-time (Appeal Brief, page 22). Yet, Applicant argues that Dubey does not teach "determining the desirability of performing at least one cache transaction based on the probability that the at least one statement will execute." A cache miss occurs when an instruction or data object needed for execution is not present in the cache memory and needs to be fetched from the slower main memory. When the compiler analyzes a program and identifies a location within the

program where a cache miss is likely to occur, it is because the compiler has determined that it is likely that an instruction requiring a cached object (data or instruction) will execute and that the needed object will not be present in the cache. The compiler inserts STOUCH prefetch instruction at an identified prefetch point because it was deemed desirable for the identified object to be cached.

Applicant argues that "with [Dubey's] process, when a given statement is detected, there is no determination as to the probability that the statement will execute. In contrast it is assumed that the statement will execute (i.e., speculative prefetch) and the STOUCH instruction for the given statement is generated to initiate a speculative prefetch and specify compile time conditions that are evaluated against the run-time outcomes to determine whether to discard or maintain the prefetch instruction/data" (Appeal Brief, page 22). However, Applicant fails provide any explanation or evidence to support the assertion that Dubey's method simply assumes that the statement will execute. Applicant admits that Dubey teaches a compiler that identifies locations where cache misses are **likely** to occur (Appeal Brief, page 20, second paragraph). The word "likely" denotes probabilities, not assumptions. Applicant also admits that Dubey teaches **speculative** prefetch. A speculation is not an assumption. Dubey's speculation is based on the likelihood of a cache miss. As explained above cache misses can only occur if a statement requiring data object or instruction to be retrieved is executed when the required item is not present in cache. Therefore, the likelihood of a cache miss is directly related to the probability that the statement needing access to an item not in cache will execute. Applicant's discussion of STOUCH instruction itself relates to Dubey's improvements over

conventional speculative prefetches, and not to the assessment or determination of the likelihood of a cache miss occurring.

Cytron Reference

In spite of explanations contained in the statement of rejections of how Cytron's disclosure anticipates "analyzing program code to determine if there is at least one statement which can affect a desirability of performing at least one cache transaction, if the at least one statement is executed" limitation, Applicant merely offers repetitive allegation that the rejection does not explain how Cytron's disclosure is even remotely related to the limitation. First of all, the Examiner notes that Cytron's disclosure of the algorithm that determines the cacheability of variables (page 231, 2.0 Algorithms) is enough to meet this limitation. Cytron discloses a three-step algorithm that identifies all variables and marks them as "cacheable", "temporarily cacheable", or "non-cacheable". These steps identify the desirability of caching each of the variables if the statement containing the variable is executed. A variable marked as "cacheable" is highly desirable to cache (see the definition of "cacheable" at page 230, left column). A statement containing a "cacheable" object or variable affects the desirability of conducting a caching transaction, because it contains an object that can always be cached. A statement containing a "non-cacheable" object also affects the desirability of performing a cache transaction because such a variable should never be cached.

Cytron discloses "determining a probability that the at least one statement will execute", because Cytron's execution model mainly concerns the analysis of Do loops for parallelizing operations in a parallel computing environment (page 230, 1.2 Execution Model). In a Do loop,

every statement is executed. Therefore, each time a Do loop is encountered, the probability of the statements in the loop being executed is 1 and the compiler analyzes the code to insert caching operation instructions based on this certainty (or probability of that is determine to be 1).

The limitation "determining the desirability of performing the at least one cache transaction based on the probability that the at least one statement will execute" is also met because when a variable referred to in a statement in a Do loop is analyzed, it is categorized into one of the three possible cacheability categories (or degrees of desirability).

Cytron, also discloses the claim limitation in another way. Cytron's analyzer analyzes program code (see for example, page 233, Figure 8; see also Figures 3 and 5) and augments the code by inserting cache transaction operations such as POST (posting of a cache value to a global memory) and IVALID (invalidation of a cache entry). The analyzer insert these cache management instructions because it has determined that it is desirable to conduct these cache transactions based on the statements in the Do loop for which the probability of execution is 1.

Rejections under 35 U.S.C. 103

While Applicant has argued 35 U.S.C 103 rejections of claims 9 and 26 under a separate heading, there is no new argument. Applicant's arguments are based on the allegation that Cytron does not anticipate the parent claims. Therefore, claims 9 and 26 stand or fall with their parent claims.

Conclusion

Any inquiry concerning this Office action should be directed to the Examiner by phone at (571) 272-4214.

Any response to this Office action should be labeled appropriately (including serial number, Art Unit 2189, and type of response) and mailed to Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, hand-carried or delivered to the Customer Service Window at the Randolph Building, 401 Dulany Street, Alexandria, VA 22313, or faxed to (571) 273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197. If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Kevin Verbrugge/

Kevin Verbrugge
Primary Examiner
Art Unit 2189

/Reginald G. Bragdon/
Supervisory Patent Examiner, Art Unit 2189